EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	2	("lazy near deletion" or (garbage near2 collect$4)) same ((tag$4 or set$4) with (creation near2 time))	USPAT	OR	OFF	2007/06/09 16:30
L2	2	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3) with (tag$3 adj (record$1 or data or object$1))))	USPAT	OR	OFF	2007/06/09 16:48
L3	13	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3) with ((tag$3 or mark$3) adj (record$1 or data or object$1))))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:41
L4	1	3 and 707/3,4,10,205,206.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:38
L5	0	3 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:41
L6	13	"lazy deletion"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:41
L7	7	6 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:50
L8	0	7 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:49

EAST Search History

L9	0	7 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language" or table$1)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:46
L10	34	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3 or eliminat$5 or ban$4 or except$5) with ((tag$3 or mark$3 or check$3) adj (record$1 or data or object$1))))	USPAT	OR	OFF	2007/06/09 16:53
L11	1163	"34" and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:50
L12	14	10 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:50
L13	0	12 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:52
L14	0	12 and (relational or SQL or "structure query l=anguage")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:52
L15	0	12 and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:52

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	2	("lazy near deletion" or (garbage near2 collect$4)) same ((tag$4 or set$4) with (creation near2 time))	USPAT	OR	OFF	2007/06/09 16:30
L2	2	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3) with (tag$3 adj (record$1 or data or object$1))))	USPAT	OR	OFF	2007/06/09 16:48
L3	13	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3) with ((tag$3 or mark$3) adj (record$1 or data or object$1))))	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:41
L4	1	3 and 707/3,4,10,205,206.ccls.	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:38
L5	0	3 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:41
L6	13	"lazy deletion"	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:41
L7	7	6 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:50
L8	0	7 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:49

EAST Search History

L9	0	7 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language" or table$1)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:46
L10	34	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3 or eliminat$5 or ban$4 or except$5) with ((tag$3 or mark$3 or check$3) adj (record$1 or data or object$1))))	USPAT	OR	OFF	2007/06/09 16:57
L12	14	10 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:58
L13	0	12 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:56
L14	0	12 and (relational or SQL or "structure query l=anguage")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:52
L15	0	12 and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:52
L16	0	10 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:58

EAST Search History

L17	43	((search$3 or quer$3 or request$3 or retriev$3) with ((exclud$3 or avoid$3 or eliminat$5 or ban$4 or except$5) with ((tag$3 or mark$3 or check$3) adj (record$1 or document$1 or data or object$1))))	USPAT	OR	OFF	2007/06/09 16:57
L18	21	17 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 17:00
L20	0	18 and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:58
L21	100	((garbage near2 collect$5) or "lazy near2 delet$4") and ("LDAP" or "lightweight directory access control protocol") and (relational or SQL or "structure query language")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 16:59
L22	0	21 and (@ad<"19990429" or @rlad<"19990429")	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/06/09 17:00

Terms used
lazy deletion or **garbage collection** and **LDAP** and **relational**

Found 40,064 of 203,282

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Formal semantics and static analysis: Compiler optimizations for nondeferred reference: counting garbage collection

Pramod G. Joisha

June 2006 **Proceedings of the 2006 international symposium on Memory management ISMM '06**

Publisher: ACM Press

Full text available: pdf(220.00 KB) Additional Information: full citation, abstract, references, index terms

Reference counting is a well-known technique for automatic memory management, offering unique advantages over other forms of garbage collection. However, on account of the high costs associated with the maintenance of up-to-date tallies of references from the stack, deferred variants are typically used in modern implementations. This partially sacrifices some of the benefits of non-deferred reference-counting (RC) garbage collection, like the immediate reclamation of garbage and short collector...

Keywords: reference counting, static analyses

2 Garbage collecting the Internet: a survey of distributed garbage collection

Saleh E. Abdullahi, Graem A. Ringwood

September 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 3

Publisher: ACM Press

Full text available: pdf(337.65 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Internet programming languages such as Java present new challenges to garbage-collection design. The spectrum of garbage-collection schema for linked structures distributed over a network are reviewed here. Distributed garbage collectors are classified first because they evolved from single-address-space collectors. This taxonomy is used as a framework to explore distribution issues: locality of action, communication overhead and indeterministic communication latency.

Keywords: automatic storage reclamation, distributed, distributed file systems, distributed memories, distributed object-oriented management, memory management, network communication, object-oriented databases, reference counting

3

The derivation of distributed termination detection algorithms from garbage collection schemes

Gerard Tel, Friedemann Mattern

January 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 15 Issue 1

Publisher: ACM Press

Full text available: pdf(2.36 MB) Additional Information: full citation, abstract, references, citings, index terms

It is shown that the termination detection problem for distributed computations can be modeled as an instance of the garbage collection problem. Consequently, algorithms for the termination detection problem are obtained by applying transformations to garbage collection algorithms. The transformation can be applied to collectors of the "mark-and-sweep" type as well as to reference-counting protocol of Lermen and Maurer, the weighted-reference-counting protocol, the local-referen ...

Keywords: distributed algorithms, distributed termination detection, garbage collection, program transformations

4 Providing high availability using lazy replication

Rivka Ladin, Barbara Liskov, Liuba Shrira, Sanjay Ghemawat

November 1992 **ACM Transactions on Computer Systems (TOCS)**, Volume 10 Issue 4

Publisher: ACM Press

Full text available: pdf(2.46 MB) Additional Information: full citation, abstract, references, citings, index terms, review

To provide high availability for services such as mail or bulletin boards, data must be replicated. One way to guarantee consistency of replicated data is to force service operations to occur in the same order at all sites, but this approach is expensive. For some applications a weaker causal operation order can preserve consistency while providing better performance. This paper describes a new way of implementing causal operations. Our technique also supports two other kinds of operations: ...

Keywords: client/server architecture, fault tolerance, group communication, high availability, operation ordering, replication, scalability, semantics of application

5 Collecting more garbage

Pascal Fradet

July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming LFP '94**, Volume VII Issue 3

Publisher: ACM Press

Full text available: pdf(1.07 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a method, adapted to polymorphically typed functional languages, to detect and collect more garbage than existing GCs. It can be applied to strict or lazy higher order languages and to several garbage collection schemes. Our GC exploits the information on utility of arguments provided by polymorphic types of functions. It is able to detect garbage that is still referenced from the stack and may collect useless parts of otherwise useful data structures. We show how to partially co ...

6 Quantifying the performance of garbage collection vs. explicit memory management

Matthew Hertz, Emery D. Berger

October 2005 **ACM SIGPLAN Notices , Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and**

applications OOPSLA '05, Volume 40 Issue 10
Publisher: ACM Press

Full text available: pdf(1.51 MB) Additional Information: full citation, abstract, references, citings, index terms

Garbage collection yields numerous software engineering benefits, but its quantitative impact on performance remains elusive. One can compare the cost of *conservative* garbage collection to explicit memory management in C/C++ programs by linking in an appropriate collector. This kind of direct comparison is not possible for languages designed for garbage collection (e.g., Java), because programs in these languages naturally do not contain calls to free. Thus, the actual gap between the tim ...

Keywords: explicit memory management, garbage collection, oracular memory management, paging, performance analysis, throughput, time-space tradeoff

7 Highly available distributed services and fault-tolerant distributed garbage collection
Barbara Liskov, Rivka Ladin
November 1986 **Proceedings of the fifth annual ACM symposium on Principles of distributed computing PODC '86**
Publisher: ACM Press
Full text available: pdf(949.55 KB) Additional Information: full citation, references, citings, index terms

8 An implementation of complete, asynchronous, distributed garbage collection
Fabrice Le Fessant, Ian Piumarta, Marc Shapiro
May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation PLDI '98**, Volume 33 Issue 5
Publisher: ACM Press

Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, citings, index terms

Most existing reference-based distributed object systems include some kind of acyclic garbage collection, but fail to provide acceptable collection of cyclic garbage. Those that do provide such GC currently suffer from one or more problems: synchronous operation, the need for expensive global consensus or termination algorithms, susceptibility to communication problems, or an algorithm that does not scale. We present a simple, complete, fault-tolerant, asynchronous extension to the (acyclic) cle ...

Keywords: distributed object systems, garbage collection, reference tracking, storage management

9 Garbage collection in object-oriented databases using transactional cyclic reference counting
P. Roy, S. Seshadri, A. Silberschatz, S. Sudarshan, S. Ashwin
August 1998 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 7 Issue 3
Publisher: Springer-Verlag New York, Inc.
Full text available: pdf(180.00 KB) Additional Information: full citation, abstract, index terms

Garbage collection is important in object-oriented databases to free the programmer from explicitly deallocating memory. In this paper, we present a garbage collection algorithm, called Transactional Cyclic Reference Counting (TCRC), for object-oriented databases. The algorithm is based on a variant of a reference-counting algorithm proposed for functional programming languages The algorithm keeps track of auxiliary reference count ...

information to detect and collect cyclic garbage. The algorithm ...

10 CONS should not CONS its arguments, or, a lazy alloc is a smart alloc

Henry G. Baker
March 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 3
Publisher: ACM Press
Full text available: pdf(1.52 MB) Additional Information: full citation, abstract, citings, index terms

Lazy allocation is a model for allocating objects on the execution stack of a high-level language which does not create dangling references. Our model provides safe transportation into the heap for objects that may survive the deallocation of the surrounding stack frame. Space for objects that do not survive the deallocation of the surrounding stack frame is reclaimed without additional effort when the stack is popped. Lazy allocation thus performs a first-level garbage collection, and if ...

11 Anatomy of LISP

John Allen
January 1978 Book
Publisher: McGraw-Hill, Inc.
Additional Information: full citation, abstract, references, cited by, index terms

This text is nominally about LISP and data structures. However, in the process it covers much broader areas of computer science. The author has long felt that the beginning student of computer science has been getting' a distorted and disjointed picture of the field. In some ways this confusion is natural; the field has been growing at such a rapid rate that few are prepared to be judged experts in all areas of the discipline. The current alternative seems to be to give a few introductory cou ...

12 An on-the-fly reference-counting garbage collector for java

Yossi Levanoni, Erez Petrank
January 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 1
Publisher: ACM Press
Full text available: pdf(787.15 KB) Additional Information: full citation, abstract, references, index terms

Reference-counting is traditionally considered unsuitable for multiprocessor systems. According to conventional wisdom, the update of reference slots and reference counts requires atomic or synchronized operations. In this work we demonstrate this is not the case by presenting a novel reference-counting algorithm suitable for a multiprocessor system that does not require any synchronized operation in its write barrier (not even a compare-and-swap type of synchronization). A second novelty of thi ...

Keywords: Programming languages, garbage collection, memory management, reference-counting

13 Garbage collection: A true hardware read barrier

Matthias Meyer
June 2006 **Proceedings of the 2006 international symposium on Memory management ISMM '06**
Publisher: ACM Press
Full text available: pdf(991.66 KB) Additional Information: full citation, abstract, references, index terms

Read barriers synchronize compacting garbage collection and application processing in a simple yet elegant way. Unfortunately, read barrier checks are expensive to implement in software, and even with hardware support, the clustering of read barrier faults irregularly impairs application progress to an unacceptable extent. For this reason, read barriers are

often considered unsuitable for hard real-time systems.In this paper, we introduce a novel hardware read barrier design for an object-based ...

Keywords: hardware support, object-based processor architecture, read barrier, real-time garbage collection

14 The space cost of lazy reference counting

Hans-J. Boehm

January 2004 **ACM SIGPLAN Notices , Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '04**, Volume 39 Issue 1

Publisher: ACM Press

Full text available: pdf(125.88 KB) Additional Information: full citation, abstract, references, index terms

Reference counting memory management is often advocated as a technique for reducing or avoiding the pauses associated with tracing garbage collection. We present some measurements to remind the reader that classic reference count implementations may in fact exhibit longer pauses than tracing collectors.We then analyze reference counting with lazy deletion, the standard technique for avoiding long pauses by deferring deletions and associated reference count decrements, usually to allocation time. ...

Keywords: garbage collection, memory allocation, reference counting, space complexity

15 Lazy replication: exploiting the semantics of distributed services

Rivka Ladin, Barbara Liskov, Liuba Shrira

August 1990 **Proceedings of the ninth annual ACM symposium on Principles of distributed computing PODC '90**

Publisher: ACM Press

Full text available: pdf(2.01 MB) Additional Information: full citation, references, citings, index terms

16 A distributed garbage collector with diffusion tree reorganisation and mobile objects

Luc Moreau

September 1998 **ACM SIGPLAN Notices , Proceedings of the third ACM SIGPLAN international conference on Functional programming ICFP '98**, Volume 34 Issue 1

Publisher: ACM Press

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a new distributed garbage collection algorithm that is able to reorganise diffusion trees and to support mobile objects. It has a modular design comprising three components: a reliable transport mechanism, a reference-counting based distributed garbage collector for non-mobile objects, and an extra layer that provides mobility. The algorithm is formalised by an abstract machine and is proved to be correct. The safety property ensures that an object may not be reclaimed as long as it i ...

17 Live-structure dataflow analysis for Prolog

Anne Mulkers, William Winsborough, Maurice Bruynooghe

March 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 2

Publisher: ACM Press

Full text available: pdf(3.59 MB) Additional Information: full citation, abstract, references, citings, index terms, review

For the class of applicative programming languages, efficient methods for reclaiming the memory occupied by released data structures constitute an important aspect of current implementations. The present article addresses the problem of memory reuse for logic programs through program analysis rather than by run-time garbage collection. The aim is to derive run-time properties that can be used at compile time to specialize the target code for a program according to a given set of queries and ...

Keywords: Prolog, abstract interpretation, compile-time garbage collection, liveness, program analysis

18 Compile-time memory reuse in logic programming languages through update in place

Gudjón Gudjónsson, William H. Winsborough
May 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 3
Publisher: ACM Press
Full text available: pdf(693.38 KB) Additional Information: full citation, abstract, references, index terms

Standard implementation techniques for single-assignment languages modify a data structure without destroying the original, which may subsequently be accessed. Instead a variant structure is created by using newly allocated cells to represent the changed portion and to replace any cell that references a newly allocated cell. The rest of the original structure is shared by the variant. The effort required to leave the original uncorrupted is unnecessary when the program will never reference ...

Keywords: Prolog, compile-time garbage collection, local reuse, reuse map, update in place

19 Embedded systems: applications, solutions, and techniques: Exploiting the efficiency of generational algorithms for hardware-supported real-time garbage collection

Sylvain Stanchina, Matthias Meyer
March 2007 **Proceedings of the 2007 ACM symposium on Applied computing SAC '07**
Publisher: ACM Press
Full text available: pdf(93.27 KB) Additional Information: full citation, abstract, references, index terms

Generational garbage collectors are more efficient than their non-generational counterparts. Unfortunately, however, generational algorithms require both write barriers and write barrier handlers and therefore degrade worst-case performance.

In this paper, we present novel hardware support for generational garbage collection. In contrast to previous work, we introduce a hardware write barrier that does not only detect inter-generational pointers, but also executes all related book-keep ...

Keywords: generational garbage collection, object-based processor architecture, real-time garbage collection, write barrier

20 Essays in computing science

C. A. R. Hoare
January 1989 Book
Publisher: Prentice-Hall, Inc.
Full text available: pdf(20.91 MB) Additional Information: full citation, abstract, references, cited by, review

Charles Antony Richard Hoare is one of the most productive and prolific computer scientists. This volume contains a selection of his published papers. There is a need, as in a Shakespearian Chorus, to offer some apology for what the book manifestly fails to

achieve. It is not a complete 'collected works'. Selection between papers of this quality is not easy and, given the book's already considerable size, some difficult decisions as to what to omit have had to be made. Pity the editor weighin ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 Next

Search Session History

Sat, 9 Jun 2007, 5:17:05 PM EST

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Search Query Display

[redacted]

Recent Search Queries

#1 (((garbage collection)<in>metadata) <and> ((ldap) <in>metadata))<and> ((relational or sql)<in>metadata)

#2 (((lazy deletion)<in>metadata) <and> ((ldap)<in>metadata)) <and> ((sql or relational)<in>metadata)

#3 (((garbage collection)<in>metadata) <and> ((ldap) <in>metadata))<and> ((sql or relational)<in>metadata)

#4 (((garbage collection)<in>metadata) <and> ((ldap) <in>metadata))<and> ((sql or relational)<in>metadata)

#5 (((lazy deletion)<in>metadata) <and> ((raltional or sql) <in>metadata))

#6 (((garbage collection)<in>metadata) <and> ((raltional or sql) <in>metadata))

[redacted]